**BLOOD BANK MANAGEMENT SYSTEM**



A Web Technologies Project Report

in partial fulfillment of the degree

**Bachelor of Technology**

in

**Computer Science & Artificial Intelligence**

**By**

**2203A51229** Yandrapati Joshith

**2203A51355** Kadidela Chanakya

**2203A51387** Sadula Manogna

**Under the Guidance of**

**Ms. K. Deepthi**

**Submitted to**



**SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE**

# SR UNIVERSITY, ANANTHASAGAR, WARANGAL

**November, 2024.**



**SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE**

**CERTIFICATE**

This is to certify that this Web Technologies Project entitled “Blood Bank Management System" is the bonafied work carried out **by Y.Joshith,K.Chanakya,S.Manogna** earing hall ticket numbers **2203A51229,2203A51355,2203A51387** for the partial fulfillment to award the degree BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE during the academic year 2024-2025 under our guidance and Supervision.

**Ms. K. Deepthi Dr. M.Sheshikala**

Assistance Professor, Professor & HOD (CSE),

SR University, SR University,Ananthasagar, Warangal. Ananthasagar, Warangal.

**External Examiner**

**ACKNOWLEDGEMENT**

We owe an enormous debt of gratitude toour Web Technologies Project guide **Ms.K.Deepthi, Assistant Professor** as well as Head of the CSE Department **Dr. M.Sheshikala, Professor** for guiding us from the beginning through the end of the Minor Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, **Dr. Indrajeet Gupta,** for his continuous support and guidance to complete this technical seminar in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

**Table of Contents**

1.Abstract 5

2. Introduction 5

3.Existing System 5

4.Proposed System -Problem Statement 6

5.Requirements 6

5.1 Functional Requirements

5.2 Non Functional Requirements

5.3 Software Requirements

5.4 Hardware Requirements

6. Design 7

7. Implementations 8

Code

8. Conclusion 12

9. Future Scope 13

10. References 13

**1. Abstract**

The Blood Bank Management System aims to efficiently manage blood donor information, blood group inventories, and donor contact details. The system leverages **Node.js**, **SQLite**, and **AngularJS** to ensure a secure, responsive, and user-friendly platform. By integrating form validations, SQL injection prevention, and a structured database, the system enhances operational efficiency for blood banks and facilitates seamless donor management.

**2. Introduction**

The availability of blood is a crucial element in saving lives, especially during medical emergencies such as accidents, surgeries, or natural disasters. In such situations, having a well-organized and efficient system for managing blood donations can significantly impact the speed and effectiveness of medical responses. However, traditional manual blood bank management systems often face numerous challenges that hinder their effectiveness. These systems are typically paper-based or rely on outdated digital methods that result in inefficiencies, human errors, and inconsistencies in maintaining records. For example, donor information may be misplaced, blood group inventories may be incorrectly tracked, or outdated contact details could lead to delays in reaching potential donors when needed.

Furthermore, traditional systems are often vulnerable to data security risks. Sensitive information such as donor contact details and blood group information may not be adequately protected, leaving the system open to data breaches, unauthorized access, and even manipulation of records. This creates not only a security concern but also undermines the integrity and trustworthiness of the blood bank's operations.

To address these issues, this project introduces a digital solution that automates and streamlines the management of blood donor information, blood group inventories, and other critical data. By digitizing the entire process, the system ensures that donor records are accurately stored, easily accessible, and securely maintained. The digital platform allows users to input and update records of blood groups, donor details, and their last donation dates, eliminating the inefficiencies and risks associated with manual record-keeping. Additionally, the system enhances the overall process by providing real-time updates, reducing the chances of errors, and improving the responsiveness of the blood bank to emergency situations.

This digital solution not only improves the operational efficiency of blood banks but also strengthens data security through modern technologies that safeguard sensitive information. With the integration of form validation, SQL injection prevention, and a structured database, the system ensures that donor data is securely stored, reducing the likelihood of human errors and malicious attacks. Ultimately, this project helps ensure a more reliable, secure, and efficient blood donation management process, contributing to better outcomes in medical emergencies.

**3. Existing System**

The traditional blood bank management system relies heavily on manual processes, including paper records and spreadsheets, leading to:

* Errors in maintaining and retrieving donor information.
* Inefficiencies in tracking blood groups and inventories.
* Difficulty in ensuring secure data storage.
* Lack of real-time updates and access.

**4. Proposed System - Problem Statement**

The **Proposed Blood Bank Management System** aims to address the limitations of the existing system by providing:

* A digital platform for storing and retrieving donor information.
* Secure handling of sensitive data, preventing unauthorized access and SQL injection attacks.
* An intuitive user interface for staff and administrators to manage records.
* Automated validations to ensure data integrity.

**5. Requirements**

**5.1 Functional Requirements**

* Add, update, and delete donor information.
* Validate donor details before storing them in the database.
* Display donor information and allow search functionality.
* Prevent SQL injection attacks using prepared statements.
* Fetch donor details in a user-friendly format.

**5.2 Non-Functional Requirements**

* **Scalability**: The system should handle increasing records and multiple concurrent users.
* **Performance**: Fetch donor information with minimal latency.
* **Security**: Protect data from unauthorized access and SQL injections.
* **Usability**: Provide a responsive, easy-to-navigate interface.

**5.3 Software Requirements**

* **Backend**: Node.js, SQLite.
* **Frontend**: AngularJS, Bootstrap.
* **Middleware**: Express.js for routing and API handling.

**5.4 Hardware Requirements**

* **Processor**: Minimum Dual-Core, 2.5 GHz.
* **RAM**: Minimum 4 GB.
* **Storage**: Minimum 1 GB for database and server files.
* **Operating System**: Windows, Linux, or macOS.

**6. Design**

A screenshot of a computer screen

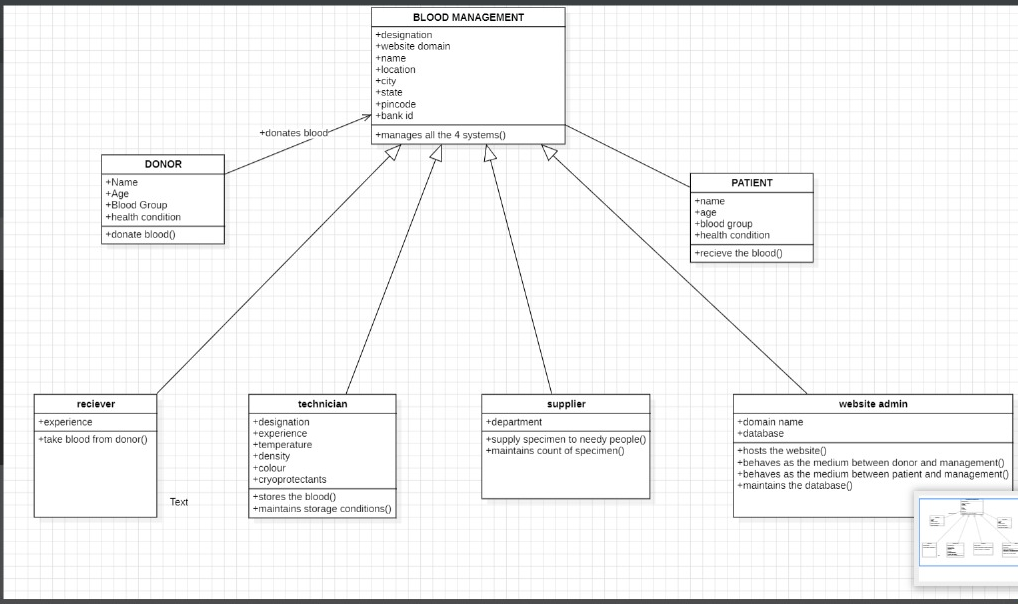
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



**A diagram of a diagram

Description automatically generated with medium confidence**

**7. Implementation**

**Code**

* **Backend**: Built using Node.js with SQLite for database management. It includes prepared statements for SQL injection prevention.

**SERVER.JS**

const express = require('express');

const sqlite3 = require('sqlite3').verbose();

const bodyParser = require('body-parser');

const cors = require('cors');

const path = require('path');

const app = express();

// Middleware

app.use(bodyParser.json());

app.use(cors());

app.use(express.static('public'));

// Database Connection

const db = new sqlite3.Database('./blood\_bank.db', (err) => {

if (err) {

console.error(err.message);

}

console.log('Connected to the SQLite database.');

});

// Routes

// Get all donors

app.get('/donors', (req, res) => {

const query = 'SELECT \* FROM blood\_bank';

db.all(query, [], (err, rows) => {

if (err) {

res.status(500).json({ error: err.message });

} else {

res.json(rows);

}

});

});

// Add a donor

app.post('/add-donor', (req, res) => {

const { donor\_name, blood\_group, contact\_number, last\_donation\_date } = req.body;

const query = `INSERT INTO blood\_bank (donor\_name, blood\_group, contact\_number, last\_donation\_date) VALUES (?, ?, ?, ?)`;

db.run(query, [donor\_name, blood\_group, contact\_number, last\_donation\_date], function(err) {

if (err) {

res.status(500).json({ error: err.message });

} else {

res.json({ message: 'Donor added successfully', id: this.lastID });

}

});

});

// Delete a donor

app.delete('/delete-donor/:id', (req, res) => {

const { id } = req.params;

const query = 'DELETE FROM blood\_bank WHERE id = ?';

db.run(query, [id], function(err) {

if (err) {

res.status(500).json({ error: err.message });

} else {

res.json({ message: 'Donor deleted successfully' });

}

});

});

// Start the server

const PORT = 3000;

app.listen(PORT, () => {

console.log(`Server running at http://localhost:${PORT}`);

});

/project-directory

/public

index.html

app.js

server.js

blood\_bank.db

* **Frontend**: AngularJS handles user input and communicates with the backend APIs using HTTP requests.

**INDEX.HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Blood Bank Admin Panel</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

<style>

/\* Add background styling \*/

body {

background-image: url('background.png'); /\* Replace with the path to your PNG image \*/

background-size: cover;

background-repeat: no-repeat;

background-attachment: fixed;

color: white; /\* Ensures text is visible on dark backgrounds \*/

}

.container {

background-color: rgba(0, 0, 0, 0.7); /\* Add a semi-transparent background for better text readability \*/

padding: 20px;

border-radius: 10px;

}

h1, h3, table, input, button {

color: white; /\* Text color to contrast the background \*/

}

input::placeholder {

color: #ddd; /\* Placeholder text color for better visibility \*/

}

</style>

</head>

<body ng-app="adminApp" ng-controller="AdminController">

<div class="container mt-5">

<h1 class="text-center">Blood Bank Admin Panel</h1>

<!-- Add Donor -->

<form class="mb-4" ng-submit="addDonor()" name="donorForm" novalidate>

<h3>Add New Donor</h3>

<div class="mb-3">

<input type="text" class="form-control" ng-model="newDonor.donor\_name" placeholder="Donor Name" required>

</div>

<div class="mb-3">

<input type="text" class="form-control" ng-model="newDonor.blood\_group" placeholder="Blood Group (e.g., A+)" required>

</div>

<div class="mb-3">

<input type="text" class="form-control" ng-model="newDonor.contact\_number"

placeholder="Contact Number" required

ng-pattern="/^\d{10}$/">

<!-- Error message for invalid contact number -->

<div class="text-danger" ng-show="donorForm.$submitted && donorForm.contact\_number.$error.pattern">

Contact number must be exactly 10 digits.

</div>

</div>

<div class="mb-3">

<input type="date" class="form-control" ng-model="newDonor.last\_donation\_date" required>

</div>

<button type="submit" class="btn btn-primary" ng-disabled="donorForm.$invalid">Add Donor</button>

</form>

<!-- Donor Table -->

<h3>Donor Records</h3>

<table class="table table-bordered">

<thead>

<tr>

<th>ID</th>

<th>Donor Name</th>

<th>Blood Group</th>

<th>Contact Number</th>

<th>Last Donation Date</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<tr ng-repeat="donor in donors track by $index">

<!-- Dynamically assign ID based on index -->

<td>{{ $index + 1 }}</td>

<td>{{ donor.donor\_name }}</td>

<td>{{ donor.blood\_group }}</td>

<td>{{ donor.contact\_number }}</td>

<td>{{ donor.last\_donation\_date }}</td>

<td>

<button class="btn btn-danger btn-sm" ng-click="deleteDonor(donor.id)">Delete</button>

</td>

</tr>

</tbody>

</table>

</div>

<script>

const app = angular.module('adminApp', []);

app.controller('AdminController', function($scope, $http) {

$scope.donors = [];

$scope.newDonor = {};

// Load all donors

$scope.loadDonors = function() {

$http.get('/donors').then((response) => {

$scope.donors = response.data;

}, (error) => {

console.error('Error fetching donors:', error);

});

};

// Add a donor

$scope.addDonor = function() {

if ($scope.donorForm.$valid) {

$http.post('/add-donor', $scope.newDonor).then((response) => {

alert(response.data.message);

$scope.loadDonors();

$scope.newDonor = {};

}, (error) => {

console.error('Error adding donor:', error);

});

}

};

// Delete a donor

$scope.deleteDonor = function(id) {

if (confirm('Are you sure you want to delete this donor?')) {

$http.delete(`/delete-donor/${id}`).then((response) => {

alert(response.data.message);

$scope.loadDonors();

}, (error) => {

console.error('Error deleting donor:', error);

});

}

};

// Initial load

$scope.loadDonors();

});

</script>

</body>

</html>

**APP.JS**

var app = angular.module('bloodBankApp', []);

app.controller('BloodBankController', function($scope, $http) {

// Initialize donor object

$scope.donor = {

donor\_name: '',

blood\_group: '',

contact\_number: '',

last\_donation\_date: ''

};

// Fetch all donors

$scope.getDonors = function() {

$http.get('/donors').then(function(response) {

$scope.donors = response.data.donors;

}, function(error) {

console.error('Error fetching donors:', error);

});

};

// Add a new donor

$scope.addDonor = function() {

$http.post('/add-blood-donor', $scope.donor).then(function(response) {

alert(response.data.message);

$scope.getDonors(); // Refresh the donor list

$scope.donor = {}; // Clear the form

}, function(error) {

alert('Error: ' + error.data.error);

});

};

// Fetch donors on load

$scope.getDonors();

});

**8. Conclusion**

The **Blood Bank Management System** offers a modern, efficient, and secure solution to address the challenges faced by traditional blood bank management systems. Traditional systems often rely on manual record-keeping or outdated technology, which can result in inefficiencies, errors, and difficulties in tracking donor information and blood inventories. For example, the process of manually recording donor details, updating blood group inventories, and contacting donors in case of urgent needs is prone to human errors, such as misplacement of data, incorrect updates, or delays in accessing vital information. Additionally, these systems are often not equipped to handle sensitive data securely, leaving them vulnerable to security breaches and unauthorized access.

This system overcomes these challenges by providing a digital, automated approach to blood bank management. It ensures that all donor information, including blood type, contact details, and last donation dates, is stored in a secure, organized database that can be accessed and updated with ease. The platform is built to be highly user-friendly, with an intuitive interface that allows staff to quickly input and retrieve donor information. This results in a significant reduction in errors and delays, enhancing the overall efficiency of blood bank operations.

Moreover, the system incorporates critical security measures to protect sensitive donor information. By implementing SQL injection prevention techniques and using a structured database, the system safeguards against malicious attacks and unauthorized data manipulation. This ensures that the integrity of the blood bank's records remains intact, providing a secure environment for both the staff and the donors.

The enhanced accessibility and accuracy of blood donation records provided by this system pave the way for smoother operations in blood banks. Real-time updates to donor data allow for quicker identification of eligible donors, better management of blood inventories, and faster responses during emergencies when blood supplies are needed urgently. By streamlining these processes, the **Blood Bank Management System** not only increases operational efficiency but also contributes to better outcomes in medical care, ensuring that blood banks can respond swiftly and accurately to life-saving needs. Ultimately, this digital solution creates a more reliable, secure, and effective platform for managing blood donations and donor information, significantly improving the overall performance of blood bank operations.

**9. Future Scope**

* **AI and Machine Learning**: Predictive analytics for blood demand, donor behavior, and inventory management to optimize donation drives and anticipate shortages.
* **Blockchain Technology**: Enhanced data security and transparency through immutable records for donor data, blood inventory, and traceability of blood products.
* **Wearable Integration**: Monitoring donors' health in real-time via wearables to assess eligibility and ensure donor safety during and after donations.
* **Global Blood Donation Network**: Connecting blood banks globally to share resources and respond to emergencies or shortages, facilitating worldwide blood distribution.
* **Gamification and Donor Engagement**: Incorporating rewards, badges, and personalized notifications to increase donor participation and retention.
* **Automated Donor Screening**: Using automated systems to assess donor eligibility based on medical data, reducing human error and improving safety.
* **Virtual Blood Donation Platforms**: Enabling remote consultation and scheduling for blood donations, especially in underserved areas.
* **Blood Donation Awareness Campaigns**: Leveraging social media for real-time promotion of donation drives and increasing public awareness of blood donation needs.

**10. References**

1. Sulaiman, S., Hamid, A. A. K. A., & Yusri, N. A. N. (2015). Development of a blood bank management system. *Procedia-Social and Behavioral Sciences*, *195*, 2008-2013.
2. Sandaruwan, P. A. J., Dolapihilla, U. D. L., Karunathilaka, D. W. N. R., Wijayaweera, W. A. D. T. L., Rankothge, W. H., & Gamage, N. D. U. (2020, December). Towards an efficient and secure blood bank management system. In *2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC)* (pp. 1-6). IEEE.
3. Ekanayaka, E. M. S. S., & Wimaladharma, C. (2015). Blood bank management system.
4. Shah, A., Shah, D., Shah, D., Chordiya, D., Doshi, N., & Dwivedi, R. (2022). Blood Bank Management and Inventory Control Database Management System. *Procedia Computer Science*, *198*, 404-409.
5. Cheema, A. S., Srivastava, S., Srivastava, P. K., & Murthy, B. K. (2015, December). A standard compliant blood bank management system with enforcing mechanism. In *2015 international conference on computing, communication and security (ICCCS)* (pp. 1-7). IEEE.
6. Babu, J. R. (2016). Blood bank management system.
7. PASHA, M. A., KANTH, M. A., & KRUPAKAR, R. (2024). Online Blood bank Management System. *International Journal of Information Technology and Computer Engineering*, *12*(2), 19-23.
8. Teena, A. C., Sankar, K., & Kannan, S. (2014). A study on blood bank management. *Middle-East Journal of Scientific Research*, *19*(8), 1123-1126.
9. Esmail, M. Y., & Osman, Y. S. H. (2018, August). Computerized Central Blood Bank Management System (CCBBMS). In *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)* (pp. 1-5). IEEE.
10. Esmail, M. Y., & Osman, Y. S. H. (2018, August). Computerized Central Blood Bank Management System (CCBBMS). In *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)* (pp. 1-5). IEEE.
11. Selvamani, K., & Rai, A. K. (2015). A novel technique for online blood bank management. *Procedia Computer Science*, *48*, 568-573.